Adaptación Dinámica de Parámetros en MCMP-SRI para el Problema de Máquina Única de Weighted Tardiness

E. de San Pedro, D. Pandolfi, A. Villagra, M. Lasso

LabTEm - Unidad Académica Caleta Olivia
Universidad Nacional de la Patagonia Austral
Caleta Olivia (9011) – Santa Cruz - Argentina
e-mail: {edesanpedro,dpandolfi,avillagra,mlasso}@uaco.unpa.edu.ar

Abstract

The adaptation of parameters and operators are one of the most important and promissory investigation areas in the evolutionary computation. The idea is adjust the algorithm to the problem while this problem is solved.

In genetic algorithms (GAs) it is not only necessary choose the representation and the operators for the problem, also we should choose values of parameters and probabilities of operators of the GA in such way that this finds the solution efficiently. The process to find "manually" appropriate values of parameters and probabilities of operators for a GA that affect significantly the performance of the algorithm, it is a task that implies a considerable consumption of time and effort. It has motivated the automation of this process.

This work analyzes the possibility to diminish the total computational effort without losing quality of the solutions. For this it intends the adaptation of the algorithm through the basic parameters that intervene in the multirecombination, like the quantity of crossovers (n1) and the quantity of parents (n2).

Keywords: multirecombination, genetic algorithms, adaptation, crossover, computational effort.

Resumen

La adaptación de parámetros y operadores es una de las más importantes y promisorias áreas de investigación en la computación evolutiva. La idea es ajustar el algoritmo al problema, mientras el problema se resuelve.

En los algoritmos genéticos (AGs) no sólo es necesario elegir la representación y los operadores para el problema, sino que también debemos elegir valores de parámetros y probabilidades de operadores del AG de manera tal que éste encuentre la solución y de manera eficiente. El proceso de encontrar "a mano" valores apropiados de parámetros y probabilidades de operadores para un AG que afecten el rendimiento del algoritmo de una manera significativa, es una tarea que implica un consumo considerable de tiempo y esfuerzo. Esto ha motivado la automatización de dicho proceso.

En el presente trabajo se analiza la posibilidad de disminuir el esfuerzo computacional total sin perder calidad de las soluciones. Para esto se propone la adaptación del algoritmo a través de los parámetros básicos que intervienen en la multirecombinación, como son la cantidad de *crossovers* (n1) y la cantidad de padres (n2).

Palabras claves: multirecombinación, algoritmos genéticos, adaptación, *crossover*, esfuerzo computacional.

1. INTRODUCCIÓN

La acción de determinar las variables y parámetros de un algoritmo genético para adecuarse a un problema, se ha llamado *adaptación del algoritmo* al problema, y en un algoritmo genético esto puede hacerse mientras el algoritmo busca una solución al problema.

Los Algoritmos Genéticos, implementan la idea de evolución y como la evolución en sí misma puede alcanzar su actual estado de sofisticación, es natural esperar que la adaptación se utilice no sólo para encontrar soluciones a un problema sino también para ajustar el algoritmo a un problema particular. Es posible modificar los parámetros durante la ejecución del AG. Ello puede hacerse usando alguna regla (posiblemente heurística), tomando información de retroalimentación del estado actual de la búsqueda o bien empleando algún mecanismo de auto-adaptación.

Es de notar que estas modificaciones pueden afectar a un elemento de un individuo (cromosoma), a todo el individuo o más aún a toda la población. Es claro que al cambiar los valores de parámetros mientras el algoritmo está buscando la solución a un problema, se gana en eficiencia.

La auto-adaptación, basada en la evolución de la evolución, fue desarrollada en las Estrategias Evolutivas para adaptar los parámetros de mutación durante la ejecución. El método ha sido extendido a otras áreas de la computación evolutiva pero todavía las representaciones, control de parámetros y operadores fijos, siguen siendo la norma.

Otras áreas de investigación se refieren a los mecanismos de adaptación, tanto a la representación de individuos, los operadores – sobre esto [16] y [17] sostienen que es claro que los operadores juegan distintos roles en distintos momentos del proceso evolutivo; los operadores deberían adaptarse -, y el control de parámetros.

En general se distinguen dos formas de fijar *valores de parámetros*:

- *Ajuste de parámetros*: Se buscan buenos valores de parámetros antes de ejecutar el algoritmo. (estático)
- *Control de parámetros*: Se ejecuta el algoritmo con valores iniciales y los mismos se van cambiando a medida que se ejecuta el algoritmo. (dinámico).

En el enfoque dinámico hay distintas maneras de realizar el control. Dichas maneras se clasifican en base a dos aspectos:

- *Cómo* trabaja el mecanismo de cambio o ajuste. El tipo de mecanismo puede ser estático o dinámico. A su vez el dinámico puede ser determinístico, adaptable o auto-adaptable.
- *Qué* componentes (operadores, selección, función de *fitness*, etc) del Algoritmo Genético se ven afectadas por el mecanismo; es decir, a qué nivel dentro del algoritmo ocurre la adaptación. Se distinguen cuatro niveles (ambiente, población, individuos, componentes).

La clasificación del tipo de adaptación se hace en base al mecanismo usado en el proceso, poniendo atención particular en el hecho de utilizar o no alguna información de retroalimentación del Algoritmo Genético [6] y [7].

- *Adaptación Estática*: los valores de los parámetros permanecen constantes durante la ejecución del algoritmo. Se necesita de un agente externo (persona o programa) para ajustar los parámetros y elegir los valores más adecuados. Generalmente se realizan numerosas corridas de prueba tratando de encontrar la relación entre los valores de parámetros y el rendimiento del algoritmo.

- *Adaptación Dinámica*: ocurre cuando existe algún mecanismo que modifica los parámetros sin control externo. La clase de algoritmos que usan el tipo de adaptación dinámica, de acuerdo al mecanismo de adaptación pueden sub-dividirse en:

    - *Adaptación dinámica determinística*: Los valores se modifican en base a una regla determinística sin usar información de retroalimentación obtenida de la ejecución del AG.

    - *Adaptación dinámica adaptable*: Se usa información de retroalimentación del algoritmo para determinar la dirección y/o magnitud del cambio de los parámetros. Ello incluye la acción de determinar si un valor se propaga o no en la población durante la evolución.

    - *Adaptación dinámica Auto-adaptable*: En este caso se usa la idea de evolución de evolución para implementar auto-adaptación de parámetros. Los parámetros a ser adaptados se codifican como parte del individuo (cromosoma) y se lo somete a los operadores genéticos recombinación y mutación.

Las siguientes secciones de este trabajo están organizadas de la siguiente manera: en la sección 2, se describe el problema de planificación abordado; el enfoque multirecombinativo utilizado (MCMP-SRI), está explicado en la sección 3; en la sección 4, se describe en detalle el mecanismo de adaptación de parámetros propuesto para reducir el esfuerzo computacional del algoritmo; en la sección 5 se muestran los detalles de implementación y los resultados obtenidos; y finalmente en la sección 6 se describen las conclusiones.

2. PROBLEMA DE PLANIFICACIÓN DE MÁQUINA ÚNICA: WEIGHTED TARDINESS

El problema de máquina única total *Weighted Tardines* [9], [15] puede ser definido de la siguiente manera: n tareas son procesadas sin interrupción en una sola máquina que puede manejar una sola tarea a la vez. La tarea j (j = 1,...,n) esta disponible para su procesamiento en tiempo cero y requiere un tiempo de procesamiento ininterrumpido ***pj*** en la máquina, una ponderación positiva ***wj***, y una fecha de entrega ***dj*** en la que idealmente debe terminarse. Para un determinado orden de procesamiento de las tareas, se puede calcular el tiempo de realización más temprano ***Cj*** y la tardanza ***Tj*** = *max {Cj-dj,0}* para la tarea j. El problema es encontrar un orden de procesamiento de las tareas con la mínima tardanza total ponderada. $\sum_{j=1}^{n} w_j T_j$

Incluso con esta formulación simple, este modelo lleva a un problema de optimización que es NP-Hard [15].

3. PROCESO DE MULTIRECOMBINACIÓN - MCMP-SRI

El proceso de multirecombinación utilizado para este trabajo, es el **MCMP-SRI** (*Multiple Crossover Multiples Parents – Stud and Random Inmigrants*) [10], [11], enfoque que ha sido extensamente probado y publicado en diferentes problemas de planificación con el cual se han obtenido siempre muy buenos resultados (para ambientes estáticos en: *Earliness* y *Tardiness* [12], *Weighted Tardiness* [3], *Average Tardiness* [13] y *Weighted Number of Tardy Jobs* [4] y para ambientes dinámicos en: *Earliness* y *Tardiness* [14], *Weighted Tardiness* [8] y *Average Tardiness* [5]), y que surge con la intención de lograr un mejor equilibrio entre la exploración y la explotación. En este enfoque, el proceso para crear los hijos es el siguiente (Ver Figura 1):

Desde la vieja población, se selecciona un individuo por medio de selección proporcional al cual se designa *stud*. El número de *n2* padres en el *pool* de apareamiento, se completa con individuos creados aleatoriamente (*random inmigrants*). El *stud* se aparea con todos los otros padres, las parejas se cruzan mediante el operador de *crossover* PMX (*Partial Mapped Crossover*) y se crean 2*(n2-1) hijos. El mejor de esos 2*(n2-1) hijos es almacenado en un *pool* de hijos temporario. La operación de *crossover* se repite *n1* veces, para diferentes puntos de corte cada vez, hasta que se complete el *pool* de hijos. Los hijos están expuestos a la mutación a través del mecanismo de intercambio de la posición. Finalmente, el mejor hijo creado desde n2 padres y con n1 *crossover*, es insertado en la nueva población.

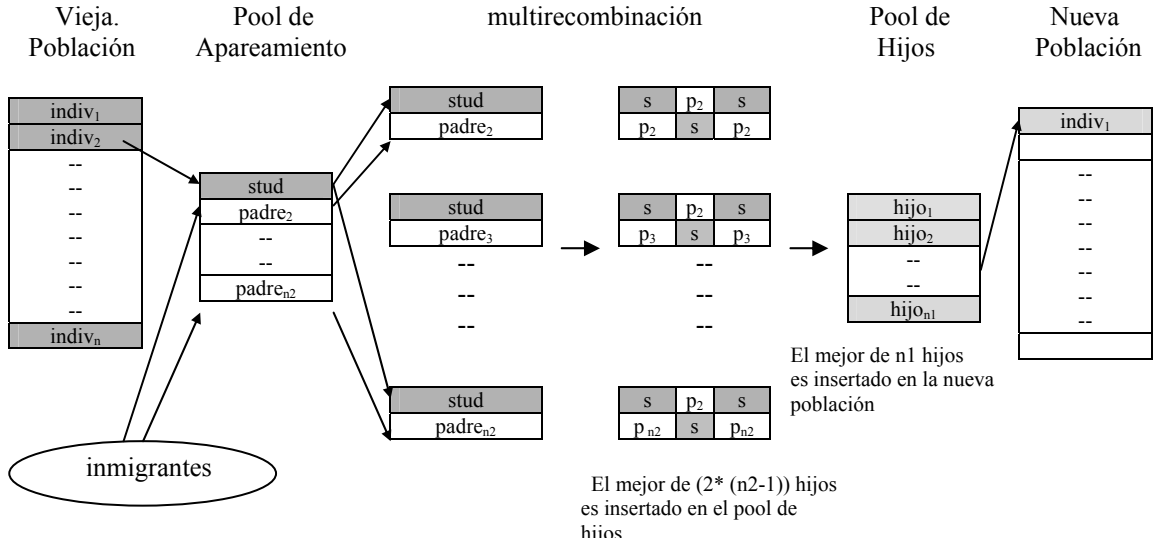

Figura 1. Proceso de multirecombinación *Stud and Random Immigrants*

4. MECANISMO DE ADAPTACIÓN PROPUESTO

El mecanismo de adaptación de los parámetros *n1* y *n2* propuesto para éste trabajo, está asociado con la utilización de diferentes estrategias, las cuales pueden hacer variar el valor de los parámetros bajo tres opciones: incrementar su valor, mantenerlo igual o disminuir su valor. Tanto el incremento como la disminución del valor, siempre es en una cantidad de **1** (uno). Considerando que las opciones son tres, para dos parámetros, surge un conjunto de 9 estrategias posibles ([=,=], [=,+], [=,-], [-,=], [+,=], [-,+], [-,-], [+,-], [+,+]).

Cada una de las estrategias tiene asociada una probabilidad de ser seleccionada (a través del mecanismo de selección proporcional), y que está relacionada con el *fitness* del mejor individuo

obtenido aplicando dicha estrategia. Inicialmente, todas las estrategias tienen la misma probabilidad de ser seleccionadas, ya que el valor de *fitness* se considera el del mejor individuo encontrado desde la población inicial. Para este mecanismo de adaptación, existe un rango permitido dentro del cual los parámetros *n1* y *n2* pueden variar y se establecieron entre [5, 20]. Las estrategias factibles se obtienen aplicando el mecanismo de selección tantas veces como sea necesario, en los casos donde los parámetros alcancen los límites inferior o superior de dicho rango.

Elegida una estrategia factible, se aplica la multirecombinación utilizando los valores de los parámetros *n1* y *n2* adaptados de acuerdo a la estrategia obtenida. Esta adaptación, como se detallará más adelante, puede realizarse en dos momentos diferentes dentro del algoritmo. Según estos dos momentos de la adaptación, vamos a obtener el mejor valor de *fitness* que actualizará a la estrategia utilizada, donde en un caso será el del mejor individuo obtenido para ser insertado en la nueva población, y en el otro caso será el mejor individuo de la nueva población generada. A continuación se muestran detalles del algoritmo utilizado, donde *maxgen* es el número de generaciones y *maxpop* el tamaño de la población:

```
procedure MCMP-SRI-Adap
    InicializaPoblacion (pop), InicializaEstrategias (est), InicializaParametros (n1, n2)
                    /* Inicializa diferente para los distintos experimentos, los valores de n1 y n2
    gen ← 1
    while (gen < maxgen) do
        ind ← 1
        while (ind <= maxpop) do
            SeleccionaEstrategia (est)
            n1, n2 ← ActualizaParametros
            pop' ← SeleccionaStud (pop)
            pop' ← GeneraInmigrantesAleatorios(n2-1)
            cantpadres ← 1
            while ( cantpadres <= n2-1 ) do
                cantcross ← 1
                while (cantcross <= n1) do
                        pop''← Recombinacion(pop')
                        pop''' ← Mutacion(pop'')
                        cantcross ← cantcross + 1
                 end-while
                 EvaluaPoblacion(pop''')
                 pool ← MejorDePoblacion (pop''')
                 cantpadres ← cantpadres + 1
                 if Adapt1 then
                     est' ← ActualizaEstrategia (est, fitnessMejorHijo)
            end-while
            EvaluaHijos(pool)
            newpop ← MejorDeHijos (pool)
            ind ← ind + 1
            if Adapt2 then
                est' ← ActualizaEstrategia (est, fitnessMejorHijo)
        end-while
        pop ← newpop
        gen ← gen + 1
    end-while
end-procedure
```

5. EXPERIMENTOS Y RESULTADOS

Durante la evolución del algoritmo se realiza una adaptación dinámica de los parámetros *n1* y *n2*, cantidad de *crossover* y cantidad de padres respectivamente, cuyos valores son esenciales en el proceso de la multirecombinación.

Se realizaron dos grupos de experimentos los cuales se diferencian entre si, en el momento de la evolución en donde se realiza la adaptación de los parámetros *n1* y *n2*. Dentro de éstos dos grupos, se realizaron corridas considerando distintos valores iniciales para los parámetros *n1* y *n2*.

Los resultados que se obtuvieron de estos experimentos, se compararon con los resultados obtenidos en un trabajo realizado anteriormente [2], donde el algoritmo fue ejecutado sin adaptación y considerando valores fijos durante toda la evolución para los parámetros *n1* y *n2*, en 18 y 20 respectivamente. En este trabajo de referencia, se adoptaron estos valores de parámetros después de realizar varias corridas con distintos valores para *n1* y *n2*, y arribar a que el algoritmo brindó los mejores resultados con esos valores (*n1 = 18 y n2 = 20*).

Resumiendo, los experimentos fueron los siguientes:

- ***SinAdap:*** Sin adaptación de parámetros, con *n1* = 18 y *n2* = 20.
- ***Adap1:*** *Adaptación cada vez que se genera un individuo para la nueva población*
    - *Exp 1:* considera n1 y n2 iniciales, igual a 5.
    - *Exp 2:* considera n1 y n2 iniciales generados aleatoriamente.
    - *Exp 3:* considera n1 y n2 iniciales, igual a 20.
- ***Adap2:*** *Adaptación cada vez que se inicia una nueva generación*
    - *Exp 4:* considera n1 y n2 iniciales, igual a 5.
    - *Exp 5:* considera n1 y n2 iniciales generados aleatoriamente.
    - *Exp 6:* considera n1 y n2 iniciales, igual a 20.

Los algoritmos evolutivos se probaron para las cinco instancias más duras (19, 41, 46, 56 y 116) de 40 tareas, para el problema de planificación en máquina única de *weighted tardiness,* seleccionados de la OR-Library [1]. Se realizaron series de 20 corridas para cada una de las instancias, y para cada uno de los experimentos. El número máximo de generaciones se fijó en 500, y el tamaño de la población se fijó en 15 individuos. Las probabilidades de *crossover* y mutación, se fijaron en 0.65 y 0.05 respectivamente, en todos los experimentos. El valor de *n1 crossovers* y de *n2* padres, varían para cada uno de los experimentos como se indicó en el punto anterior.

Para comparar los algoritmos, se eligieron las siguientes variables de rendimiento relevantes:

Ebest = ((mejor valor - opt_val)/opt_val)*100. Este es el error porcentual del mejor individuo encontrado, comparado con el valor óptimo conocido o estimado opt_val. Esto nos da una medida sobre cuán lejos está el mejor individuo del opt_val.

Gbest. Ésta es la generación donde se ha encontrado el mejor individuo.

Evals: Es el número de evaluaciones necesarias para obtener al mejor individuo en una corrida.

Las siguientes tablas muestran los valores medios de las variables de rendimiento para las instancias seleccionadas, bajo los parámetros de *n1* y *n2* definidos para cada experimento. Los valores indicados en negrita reflejan el mejor rendimiento para cada instancia. Al final de cada tabla se indican los valores medios del promedio (*Avg*), mínimo (*Min*) y máximo (*Max*), de la variable de rendimiento correspondiente.

Tabla 1. *Ebest* de cada instancia y para todos los experimentos

Inst	Opt_val	SinAdap	Adapt 1			Adapt 2		
			Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
19	77122	***0,33***	0,56	0,58	0,54	0,58	0,57	***0,47***
41	57640	***0,11***	0,84	0,79	0,86	0,79	0,80	***0,64***
46	64451	***0,03***	0,26	0,28	0,24	0,44	0,23	***0,22***
56	2099	8,06	***7,76***	8,25	8,67	9,04	7,80	8,94
116	46770	***0,33***	1,35	1,34	1,23	1,92	***1,19***	0,88
	Avg	**1,77**	**2,15**	**2,25**	**2,31**	**2,56**	***2,12***	**2,23**
	Min	**0,03**	**0,26**	**0,28**	**0,24**	**0,44**	**0,23**	**0,22**
	Max	**8,06**	**7,76**	**8,25**	**8,67**	**9,04**	**7,80**	**8,94**

La Tabla 1 resume los valores medios de *Ebest* de cada una de las instancias, para los distintos experimentos. Estos resultados se muestran como el promedio de las 20 corridas realizadas para cada instancia. Aquí, puede observarse que para tres de las instancias, en el *Exp6* se obtienen soluciones de mejor calidad.

En la Tabla 2 se muestra con el fin de dejar en claro que la generación en la cual el algoritmo con adaptación dinámica encuentra al mejor individuo, siempre es mayor que el algoritmo donde no se aplicó la adaptación. Esto es una consecuencia de que al reducirse el esfuerzo computacional, el algoritmo necesita evolucionar por más generaciones para encontrar el mejor individuo.

Tabla 2. *Gbest* de cada instancia y para todos los experimentos

Inst	Opt_val	SinAdap	Adapt1			Adapt2		
			Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
19	77122	***355,2***	475,95	465,25	475,10	476,40	452,35	***441,10***
41	57640	***371,0***	***423,00***	468,25	470,00	479,30	463,45	467,60
46	64451	***371,0***	476,75	466,40	479,65	474,90	483,25	***464,30***
56	2099	***83,2***	186,35	204,15	***163,10***	243,20	189,65	197,05
116	46770	***441,0***	468,15	471,70	477,35	482,45	477,75	***459,05***
	Avg	***324,28***	**406,04**	**415,15**	**413,04**	**431,25**	**413,29**	***405,82***
	Min	**83,20**	**186,35**	**204,15**	**163,10**	**243,20**	**189,65**	**197,05**
	Max	**441,00**	**476,75**	**471,70**	**479,65**	**482,45**	**483,25**	**467,60**

Tabla 3. *Evals* de cada instancia para cada uno de los experimentos

Inst	Opt-val	SinAdap	Adapt 1			Adapt 2		
			Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
19	77122	3623040	2280035	2271023	2323921	***1884158***	2021123	2842230
41	57640	3784200	2680418	2263687	2285593	***1831977***	2222157	2425919
46	64451	3784200	2272568	2352503	2353466	***1785203***	2090126	2534681
56	2099	***848640***	2321475	2287439	2364955	***1715090***	1960614	2573576
116	46770	4498200	2269974	2341823	2321581	***1757316***	2278629	2577966
	Avg	**3307656**	**2364894**	**2303295**	**2329903**	***1794749***	**2114530**	**2590874**
	Min	**848640**	**2269974**	**2263687**	**2285593**	**1715090**	**1960614**	**2425919**
	Max	**4498200**	**2680418**	**2352503**	**2364955**	**1884158**	**2278629**	**2842230**

La Tabla3 muestra los valores medios de *Evals* de cada una de las instancias para los distintos experimentos. Aquí puede observarse claramente que para todos los experimentos, tanto en *Adap1* como en *Adapt2*, se reduce significativamente la cantidad de evaluaciones realizadas por el algoritmo, comparado con los valores sin adaptación. De estos dos grupos de experimentos, los de *Adap2* brindan los mejores resultados, y dentro de éstos el *Exp4* se comporta mejor en todas las instancias. En general, el mejor valor en promedio lo da el *Exp4* con ***1794749*** evaluaciones, mientras que el peor resultado se obtuvo del *Exp6* con un valor de ***2590874***. Este peor resultado, siempre sigue siendo mejor que el resultado obtenido sin adaptación.

Para observar mejor los resultados de la Tabla 3, se presenta el Gráfico 1, en donde se muestra una comparación del comportamiento de la variable de rendimiento *Evals* para la mejor corrida de cada una de las instancias del Exp4. Aquí, la cantidad de evaluaciones realizadas en cada generación, están siempre por debajo del valor que permanece fijo durante todas las generaciones y que corresponden a las evaluaciones que realiza el algoritmo cuando no se adaptan los parámetros *n1* y *n2*; sólo en algunas pocas generaciones éstos valores superan éste valor constante.

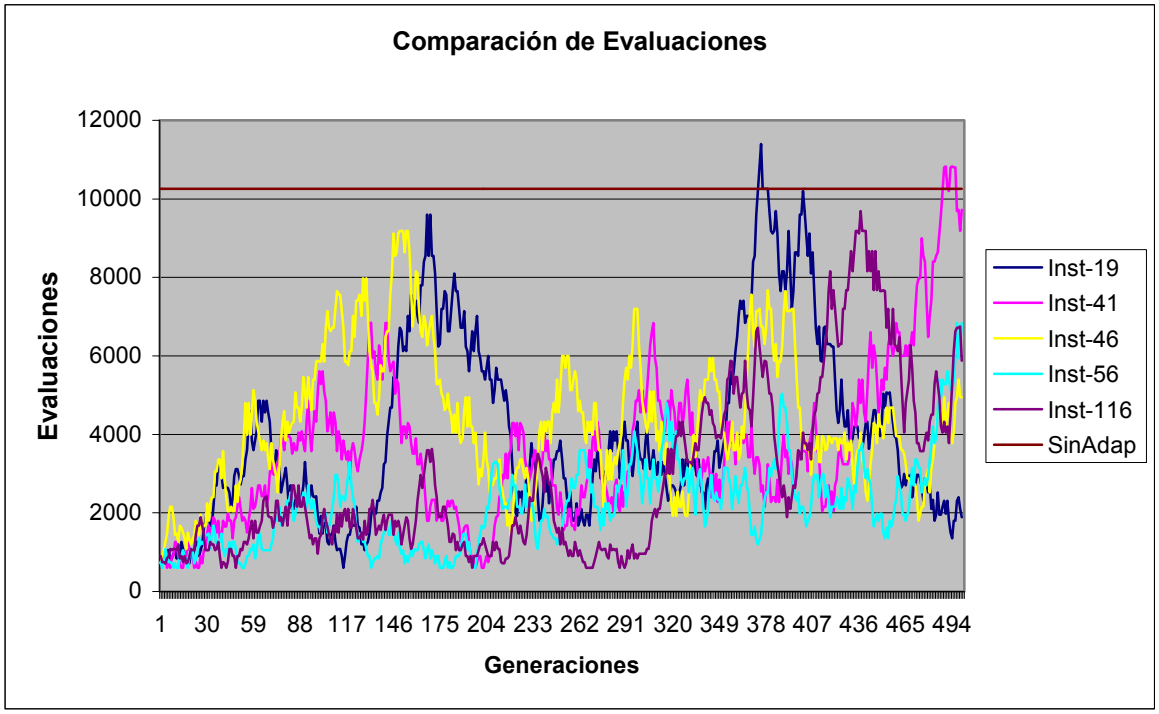

Gráfico 1. Comportamiento de *Evals* para la mejor corrida de cada instancia

Otra forma de medir los resultados obtenidos, es evaluando la calidad de las soluciones (*Ebest*) aplicando el mismo esfuerzo computacional (*Evals*) en todos los experimentos, con adaptación dinámica de parámetros y sin adaptación. La Tabla 4 muestra el error porcentual para cada instancia y para cada experimento; así la columna ***SA1*** muestra los errores porcentuales del algoritmo sin adaptación de parámetros y la columna ***A1*** con adaptación de parámetros, con la misma cantidad de evaluaciones. De igual manera se muestra para el resto de los experimentos.

En general, se puede observar que para las instancias 19, 41 y 56, todos los experimentos que utilizaron adaptación dinámica superaron en calidad al algoritmo sin adaptación, mientras que en las instancias 46 y 116, el algoritmo sin adaptación supera en calidad a las estrategias adaptativas.

Tabla 4. Comparación de Ebest con el mismo esfuerzo computacional

Inst	SA1	A1	SA2	A2	SA3	A3	SA4	A4	SA5	A5	SA6	A6
19	0,99	**0,56**	0,99	**0,58**	0,86	**0,54**	1,09	**0,58**	0,99	**0,57**	0,57	**0,47**
41	2,08	**0,84**	2,91	**0,79**	2,91	**0,86**	3,01	**0,79**	2,91	**0,80**	2,17	**0,64**
46	**0,16**	0,26	**0,16**	0,28	**0,16**	0,24	**0,19**	0,44	**0,19**	0,23	**0,16**	0,22
56	13,29	**7,76**	13,29	**8,25**	13,29	**8,67**	13,96	**9,04**	13,96	**7,80**	13,29	**8,94**
116	**0,88**	1,35	**0,83**	1,34	**0,88**	1,23	**1,76**	1,92	**0,88**	1,19	**0,67**	0,88

6. CONCLUSIONES

A lo largo de varios trabajos realizados, el objetivo principal ha sido estudiar distintos mecanismos que permitan reducir el esfuerzo computacional del algoritmo MCMP-SRI, con la mínima pérdida de la calidad en los resultados. En éste enfoque, la recombinación del semental (*stud*) y los inmigrantes aleatorios (*random immigrants*) aplicado a problemas de planificación, y con representación basada en permutaciones, recombina el material genético promovido desde individuos evolucionados (semental) y el generado aleatoriamente (inmigrantes aleatorios).

En todos estos trabajos, el algoritmo se ha sometido a una adaptación estática, ya sea insertando conocimiento específico del problema, aplicando diferentes operadores de *crossover*, variando algunos valores de parámetros: tamaño de la población, cantidad de generaciones, cantidad de padres y *crossovers*, etc. Aquí se propone someter al algoritmo a una adaptación dinámica, modificando los valores de los parámetros (*n1* y *n2*) que regulan la exploración y la explotación en los procesos de multirecombinación.

De los experimentos realizados, se pueden apreciar las siguientes consideraciones:

- En todos los experimentos realizados, se puede observar que la calidad de los resultados no se ha perdido, comparando con los resultados obtenidos del algoritmo sin adaptación, tal como se muestra en la Tabla 1, y en particular en la Tabla 4 en donde puede verse que en términos generales, con igual esfuerzo computacional la calidad de las soluciones con adaptación de parámetros, son mejores que las soluciones sin adaptación. No existe una tendencia que sobresalga sobre la otra respecto a los dos grupos de experimentos (*Adap1 ó Adap2*), y ambas obtienen resultados de calidad similar.

- Los resultados de la variable de rendimiento *Gbest*, determinan que quizás el algoritmo necesite evolucionar más para obtener mejores resultados, ya que las generaciones en las se

que encuentran las mejores soluciones están muy cercanas a la generación fijada como criterio de parada del algoritmo. Para el caso del algoritmo sin adaptación de parámetros, por cada generación se realizan un número de evaluaciones elevadas, teniendo en cuenta el valor de los parámetros *n1* y *n2* (18 y 20 respectivamente); para el algoritmo en donde se varían los valores de los parámetros, puede suceder especialmente cuando éstos valores son bajos, que el algoritmo no encuentre el mejor individuo en las primeras generaciones. En cuanto a las evaluaciones realizadas (*Evals*) por los dos grupos de experimentos (*Adapt1* y *Adapt2*), se puede observar que los del grupo *Adap2*, requieren menor esfuerzo computacional que los de *Adapt1*.

Las dos variantes presentadas para la adaptación dinámica de parámetros, reducen la cantidad de evaluaciones (*Evals*), lo que provoca la aceleración de la convergencia con una mínima pérdida de la calidad (*Ebest*), aunque no totalmente generalizable para algunas instancias.

En futuros trabajos se aplicará la adaptación dinámica de otros parámetros de control del algoritmo. Se estudiará además, en profundidad, el comportamiento de los parámetros *n1* y *n2* Se aplicarán también, éstos y otros mecanismos de adaptación, en otros problemas de planificación.

AGRADECIMIENTOS

Se reconoce a la Universidad Nacional de la Patagonia Austral por su apoyo al grupo de investigación y la cooperación y las críticas constructivas proporcionadas por el mismo.

REFERENCIAS

[1] Beasley J. E. Weighted Tardiness Scheduling, OR Library, http://mscmga.ms.ic.ac.uk/

[2] de San Pedro M., Lasso M., Villagra A., Pandolfi D., Gallard R. - *Influence of Crossover Operators in Evolutionary Scheduling Under Multirecombined Schemes* - IX Congreso Argentino de Ciencias de la Computación – CACIC 2003 pp 658-669, REDUNCI – Universidad Nacional de La Plata, La Plata (Buenos Aires) – Argentina, Octubre 2003.

[3] de San Pedro M.E., Lasso M., Villagra A., Pandolfi D., Gallard R.; *Solutions to the Dynamic Average Tardiness Problem in the Single machine Environments*; IX Congreso Argentino de Ciencias de la Computación – CACIC 2003, La Plata, Argentina, octubre, 2003, pp. 1251-1258.

[4] de San Pedro M.E., Pandolfi D., Villagra A., Vilanova G., Gallard R.; *Stud and immigrants in multirecombined evolutionary algorithm to face weighted tardiness scheduling problems*; VII Congreso Argentino de Ciencias de la Computación- CACIC 2001, El Calafate, Argentina, octubre, 2001, pp. 1251-1258.

[5] de San Pedro M.E., Villagra A., Lasso M., Pandolfi D., Diaz Vivar M., Gallard R.; *Solutions for the Weighted Number of Tardy Jobs in Single Machine Environments via Evolutionary Algorithms*; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, June 2003, Brazil, pp. 438-443.

[6] Eiben, A.E., Hinterding, R., and Michalewicz, Z., Parameter Control in Evolutionary

Algorithms, IEEE Transactions on Evolutionary Computation, Vol.3, No.2, 1999.

[7] Hinterding, R., Michalewicz, Z., and Eiben, A.E., *Adaptation in Evolutionary Computation: A Survey*, Proceedings of the 4th IEEE International Conference on Evolutionary Computation, Indianapolis, April 13-16, 1997, pp.65-69.

[8] Lasso M., Pandolfi D., De San Pedro M.E., Villagra A., Gallard R.; *Heuristics to Solve Dynamic W-T problems in Single Machine Environments*; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, June 2003, Brazil, pp. 432-437.

[9] Morton T., Pentico D., *Heuristic scheduling systems*, Wiley series in Engineering and technology management. John Wiley and Sons, INC, 1993.

[10] Pandolfi D., Lasso M., De San Pedro M.E., Villagra A., Gallard R.; *Evolutionary Algorithms to solve average tardiness problems in the single machine environments*; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, June 2003, Brazil, pp. 444-449.

[11] Pandolfi D., Vilanova G., De San Pedro M., Villagra A. *Multirecombining studs and immigrants in evolutionary algorithm to face earliness-tardiness scheduling problems*. Proceedings of the International Conference in Soft Computing. University of Paisley, Scotland, U.K., June2001.

[12] Pandolfi D., Vilanova G., De San Pedro M.E, Villagra A.; Gallard R.; *Evolutionary algorithms to minimize earliness-tardiness penalties from a common due date*; Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. III Emergent Computing and Virtual Engineering, pp. 405-408, Orlando, Florida July 2001

[13] Pandolfi D., Vilanova G., De San Pedro M.E, Villagra A.; Gallard R.; *Adaptability of multirecombined evolutionary algorithms in the single-machine common due date problem*; Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. III Emergent Computing and Virtual Engineering, pp. 401-404, Orlando, Florida July 2001.

[14] Pandolfi D.; Vilanova G.; De San Pedro M.; Villagra A. Gallard R. *Solving the Single-Machine Common Due Date Problem Via Stud and Inmigrants in Evolutionary Computation* vol III pp 409-413 World Multiconference on Systemics, Cybernetics and Informatics Orlando 2001.

[15] Pinedo, Michael.- *Scheduling: Theory, Algorithms and System*. Prentice Hall, first edition, 1995. pp 44-48, 143-145.

[16] Schaffer J.D. and Morishima A. *An adaptive crossover distribution mechanism for genetic algorithms*. In Proceedings of the 2nd International Conference on Genetic Algorithms. Laurence Erlbaum Associates, 1987. Pp, 38-40.

[17] Spears W.M. *Adapting Crossover in evolutionary algorithms*. In J.R. Mc. Donnell, R.G. Reynolds, and D.B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming,* The MIT Press, 1995, pp. 367-384.